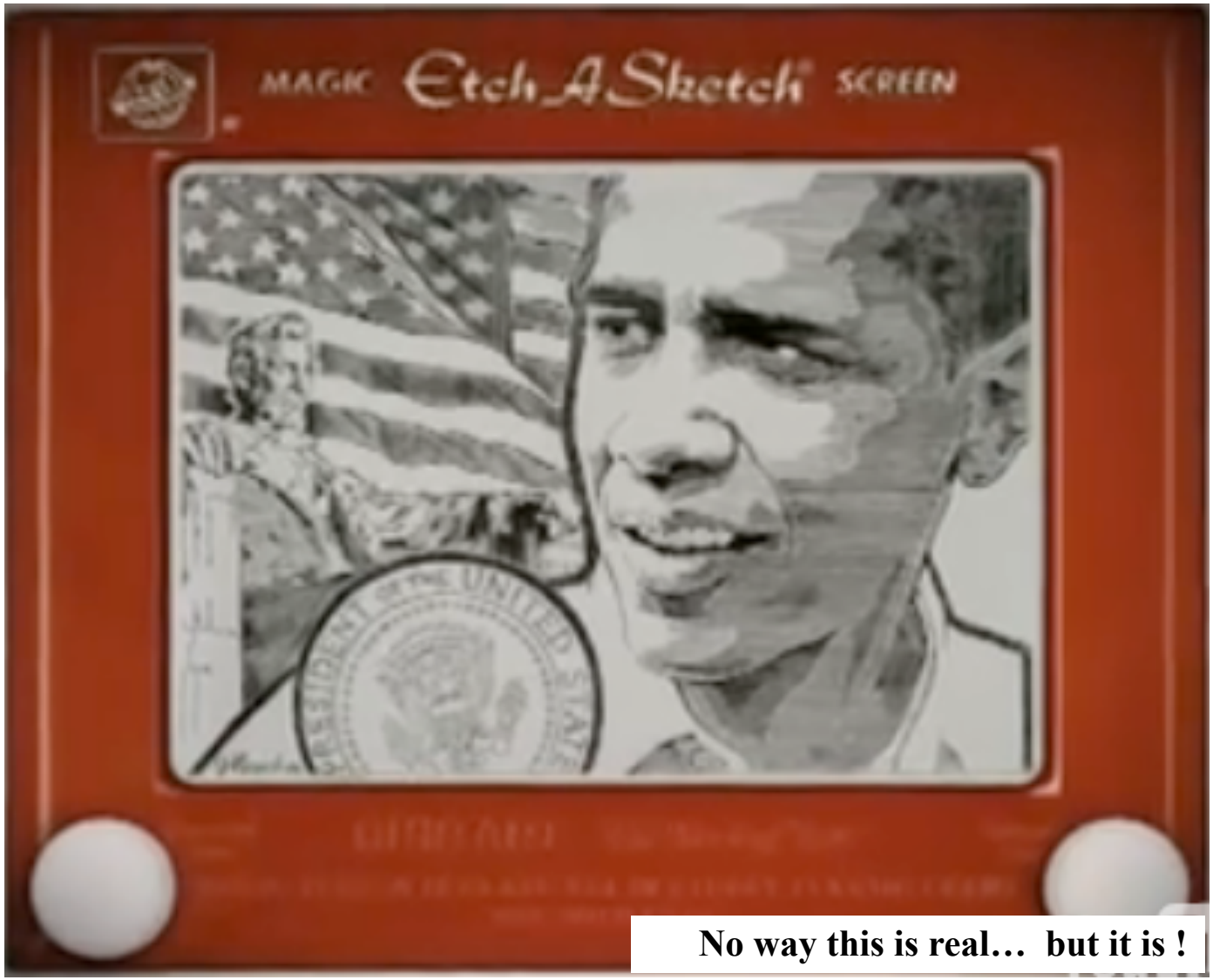
Etch-a-Sketch ?



No way this is real… but it is !

more usual etch-a-sketch work...



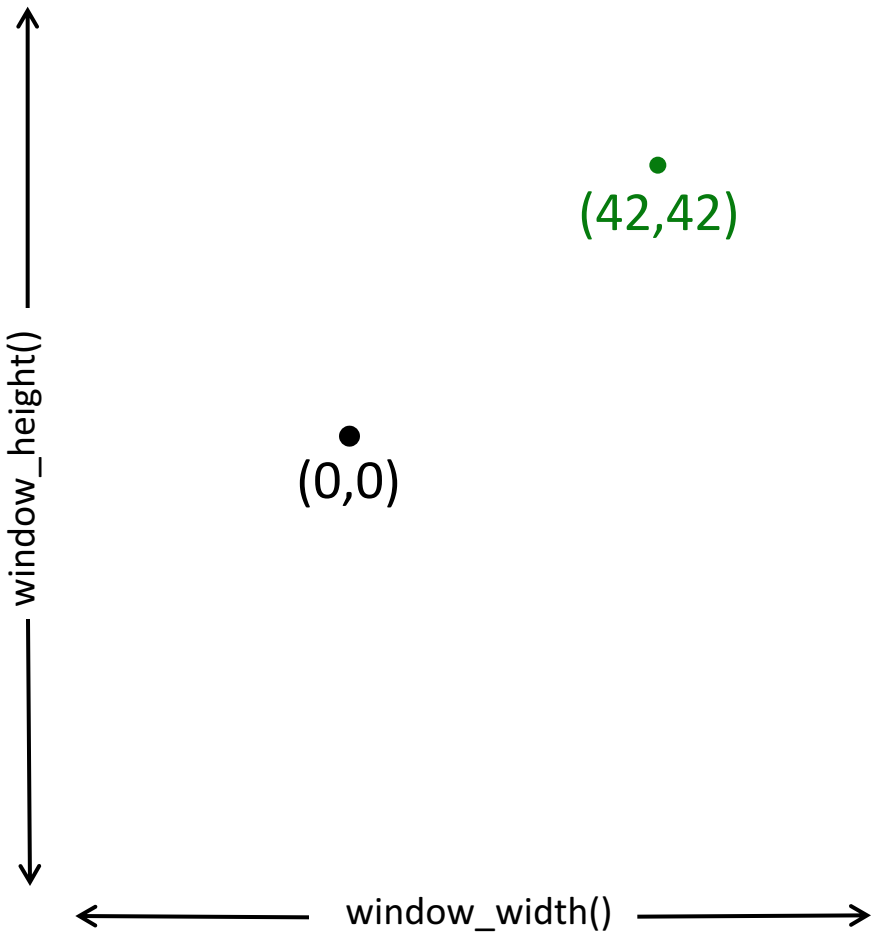
Python's Etch-a-Sketch

```
import time
from turtle import *

def draw():      # define it!
    shape('turtle')
    # pause
    time.sleep(2)
    # drawing...
    width(5)
    left(90)
    forward(50)     pixels!
    right(90)     degrees!
    backward(50)
    down() or up()     is the pen on the "paper"?
    color('darkgreen')
    tracer(1) or tracer(0)
```

sets if the pen animates or not

```
# run it and done!
draw()
done()
```



http://docs.python.org/library/turtle.html

Turtle happiness?



you can always run

```
done()
```

after turtle drawing

This releases control of the turtle window to the computer (the operating system).

don't use it in your functions, however

Single-path recursion

```
def tri():   # define it!
    """A triangle!
    """
    forward(100)
    left(120)
    forward(100)
    left(120)
    forward(100)
    left(120)

# run now
tri()
```

(1) Let's **tri** this with recursion:

```
def tri(n):
    """Draws a triangle"""
    if n == 0: return
    else:
        forward(100) # one side
        left(120)    # turn 360/3
        tri(n-1)   # draw rest
```

(2) How about ***any*** regular N-gon?

```
def poly(n, N):
    """Draws a polygon"""
    if n == 0: return
    else:
        forward(100)   # one side
        left(360 / N)  # turn 360/N
        poly(n-1, N)   # draw rest
```

Random Walk



one possible result of **rwalk(20)**

Have **rwalk** draw a "stock-market" path:

N steps of 10 pixels each. *Use recursion.*

```
from random import *

def rwalk(N):
  """Make N 10-pixel steps, NE or SE"""

  if N == 0: return

  elif choice(['left', 'right']) == 'left':
        left(45)
        forward(10)
                                  ⇦ ?

  else:  # this handles 'right'
                                  ⇦ ?
```

Extra! How could you make this a bull (or a bear) market?

```
from random import *

def rwalk(N):
    """Make N 10-px steps, NE or SE"""
    if N == 0:    return

    elif choice(['left', 'right']) == 'left':
        left(45)
        forward(10)
        right(45)
        rwalk(N - 1)

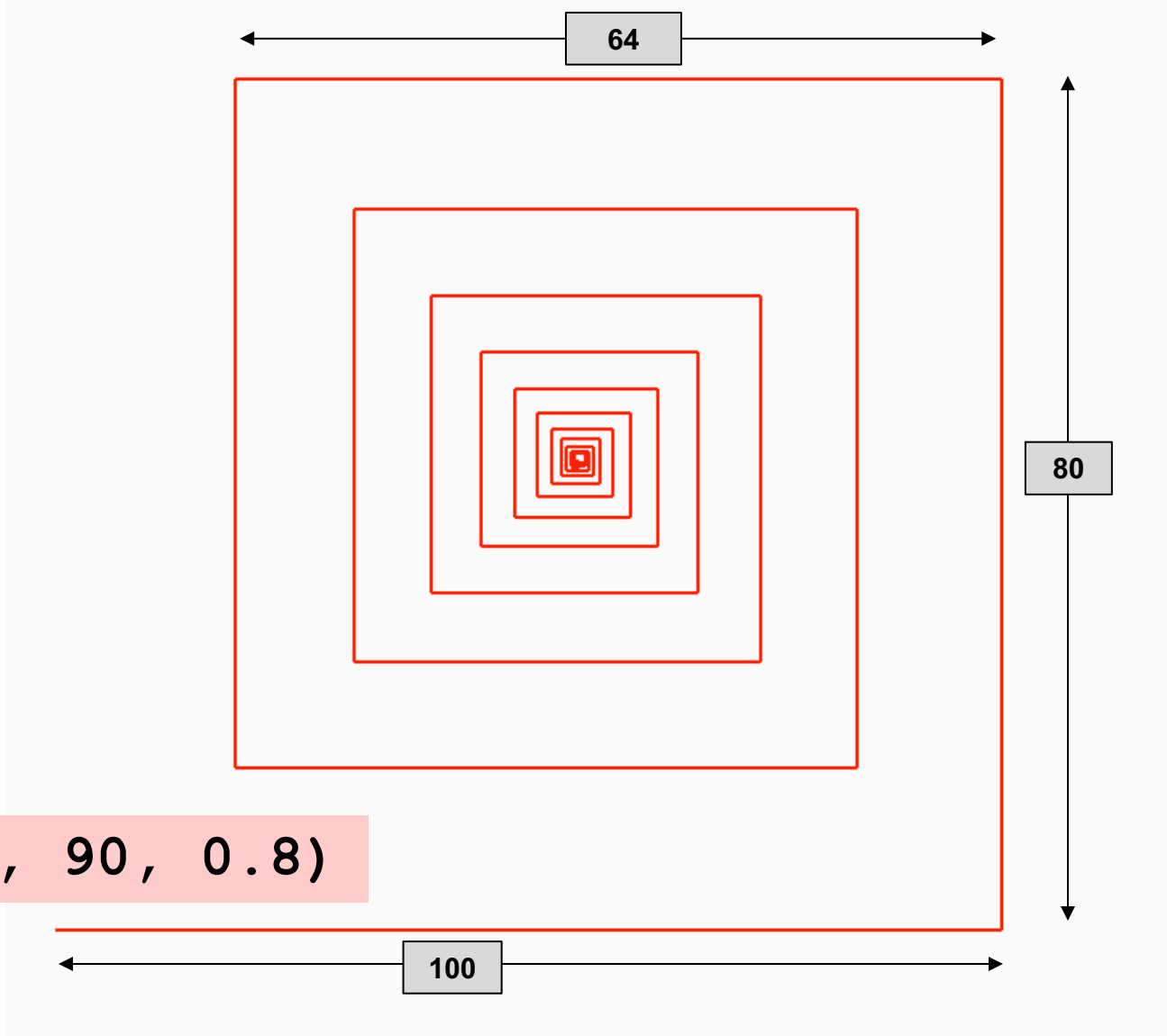
    else:  # 'right'
        right(45)
        forward(10)
        left(45)
        rwalk(N - 1)
```

rwalk(N) is a random "stock market" walk...

What if we ***didn't*** turn back to face east each time?

Single-path (or counting) recursion

Fractal Art

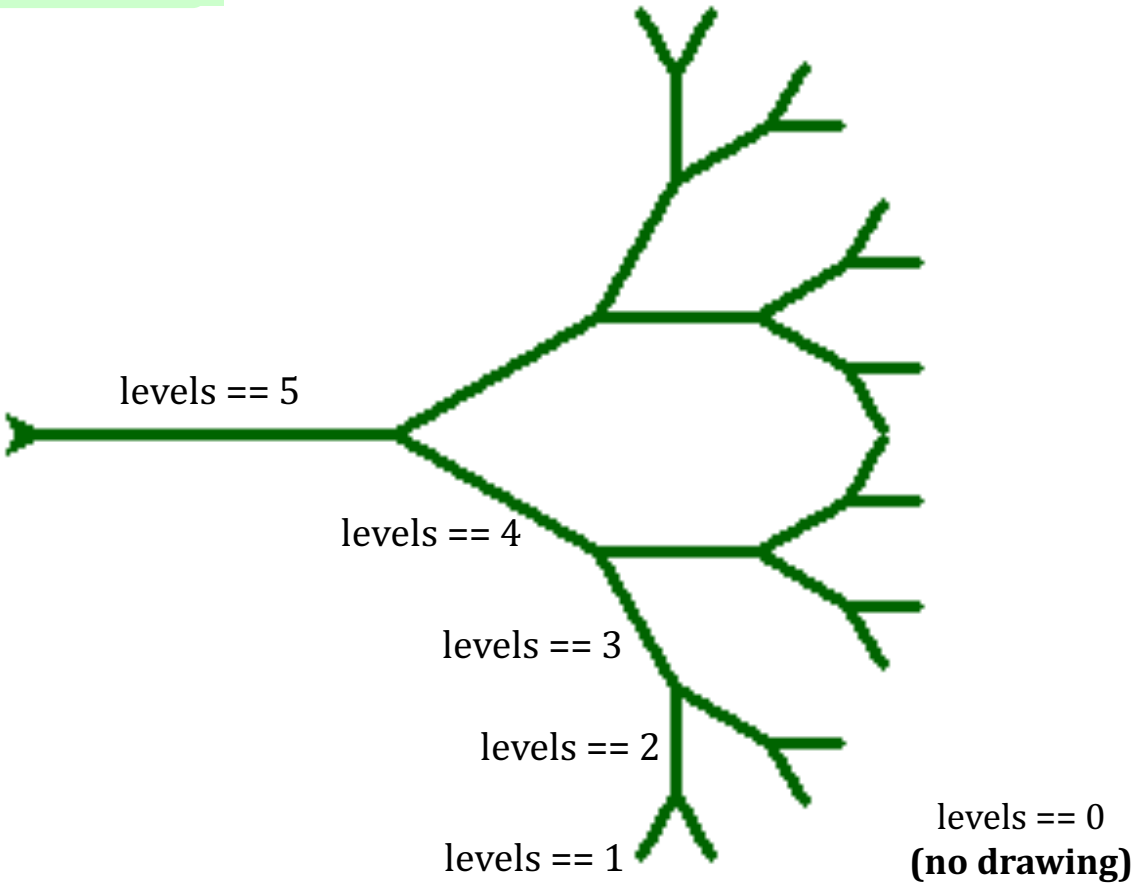



```
spiral(100, 90, 0.8)
```

```
spiral(initLength, angle, multiplier)
```

svtree(trunkLength, levels)

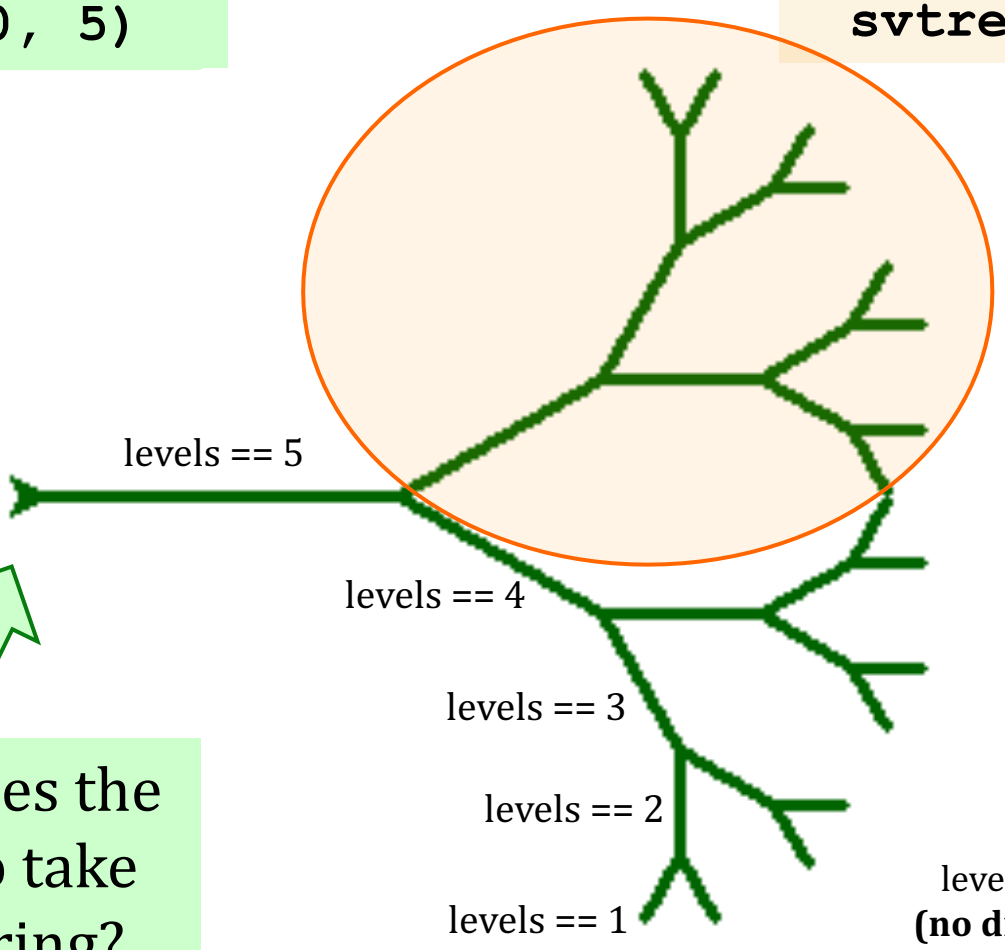
```
svtree(100, 5)
```



`svtree( trunkLength, levels )`

`svtree(100, 5)`

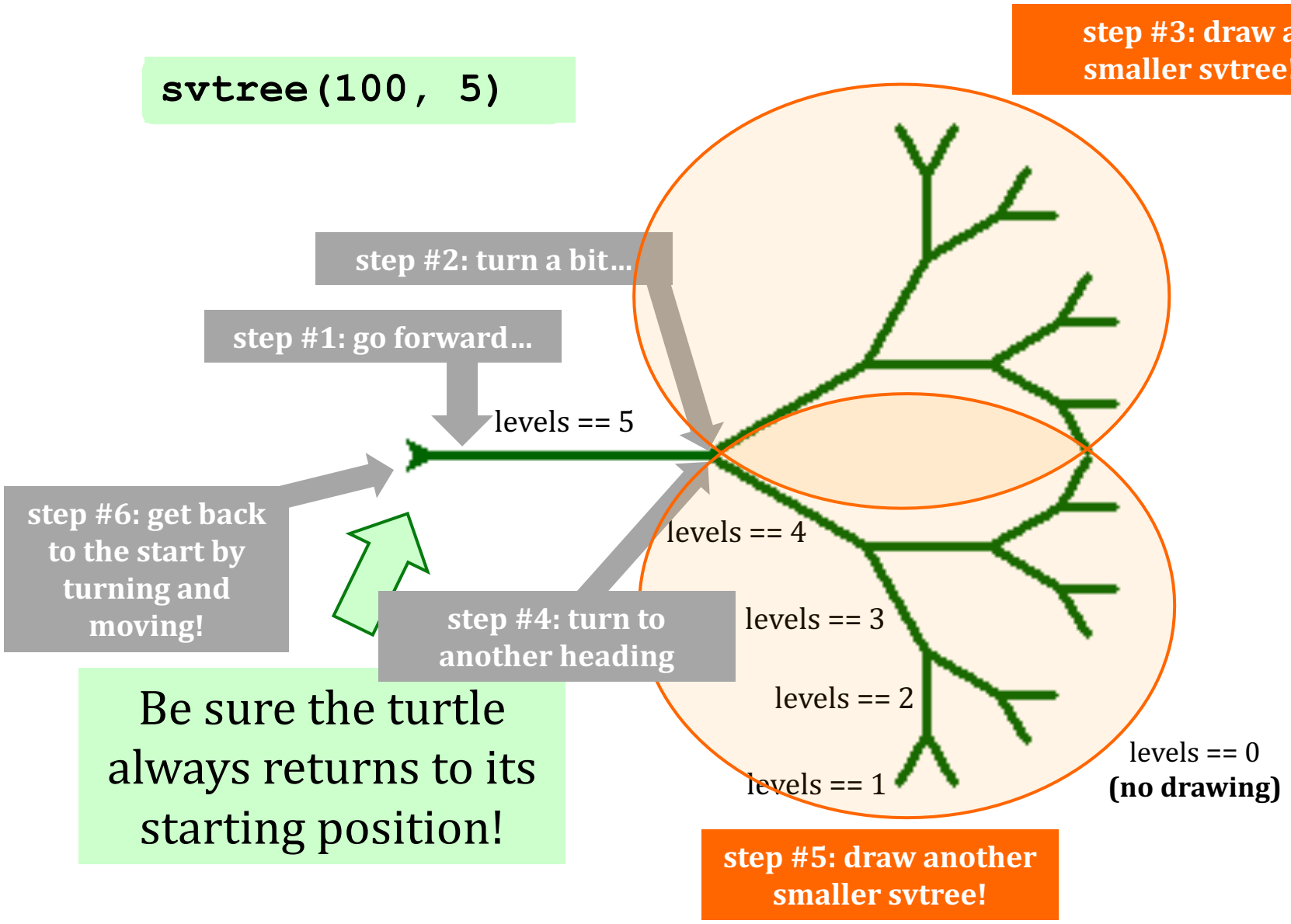
`svtree(75, 4)`



What steps does the turtle need to take before recurring?

svtree(trunkLength, levels)

svtree(100, 5)



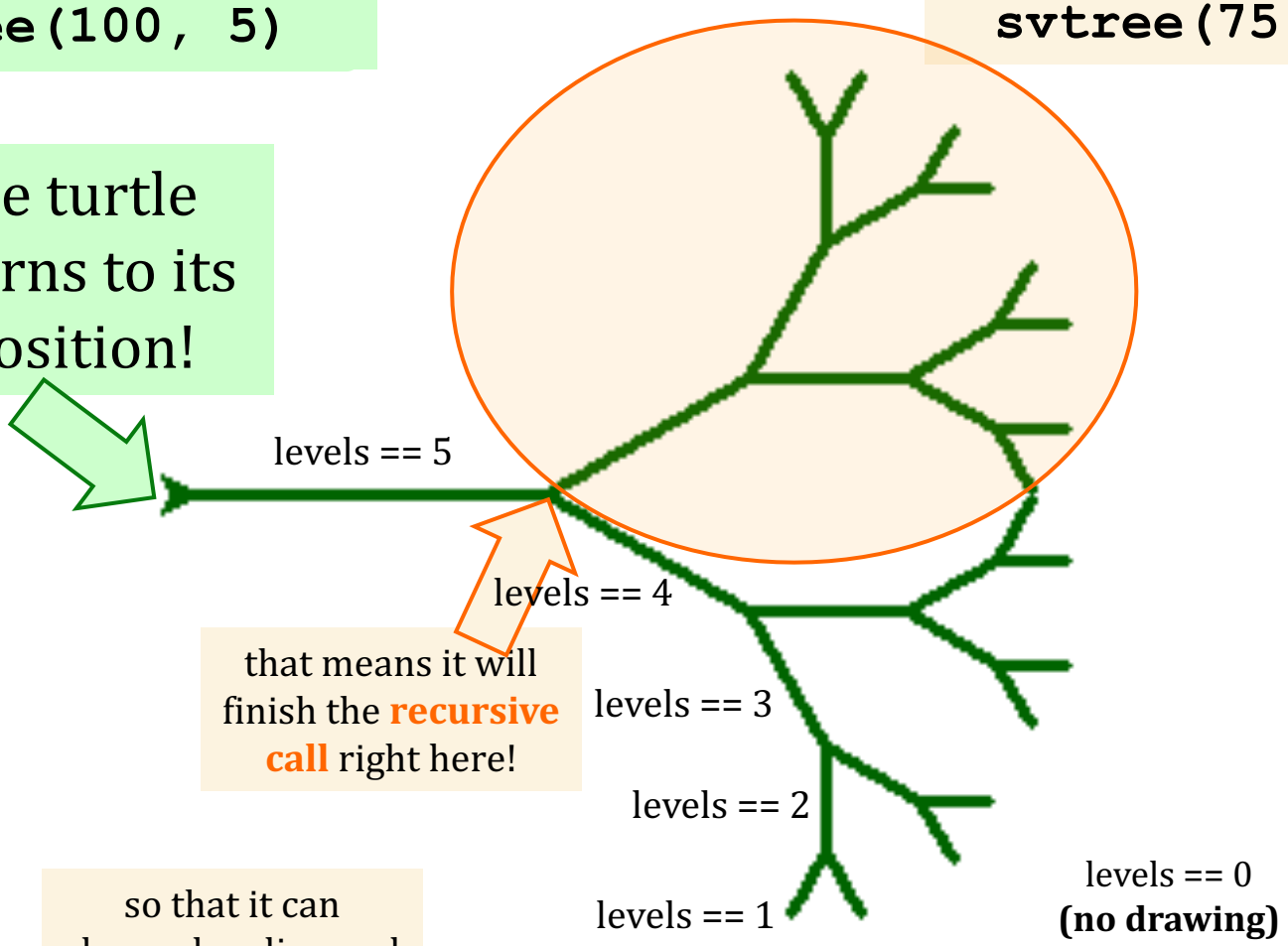
Be sure the turtle always returns to its starting position!

`svtree(trunkLength, levels)`

`svtree(100, 5)`

`svtree(75, 4)`

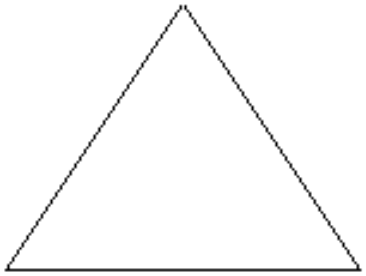
Be sure the turtle always returns to its starting position!



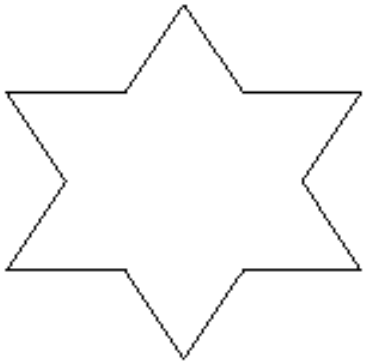
that means it will finish the **recursive call** right here!

so that it can change heading and draw another one…

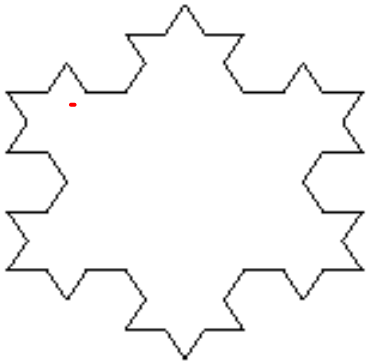
The Koch curve



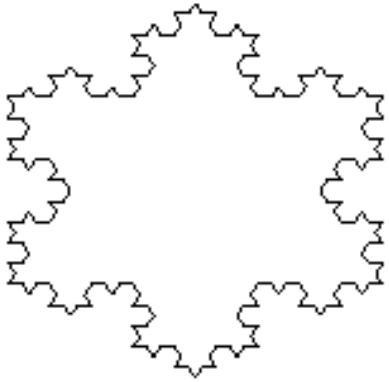
`snowflake(100, 0)`



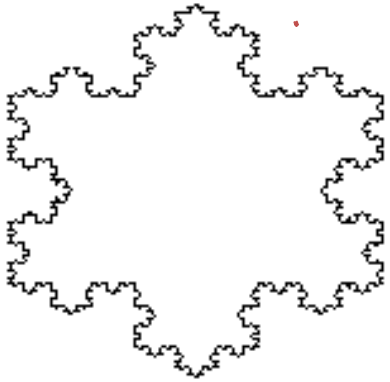
`snowflake(100, 1)`



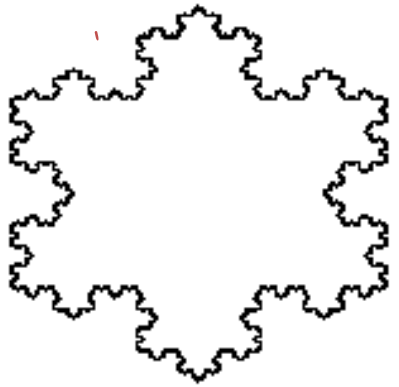
`snowflake(100, 2)`



`snowflake(100, 3)`

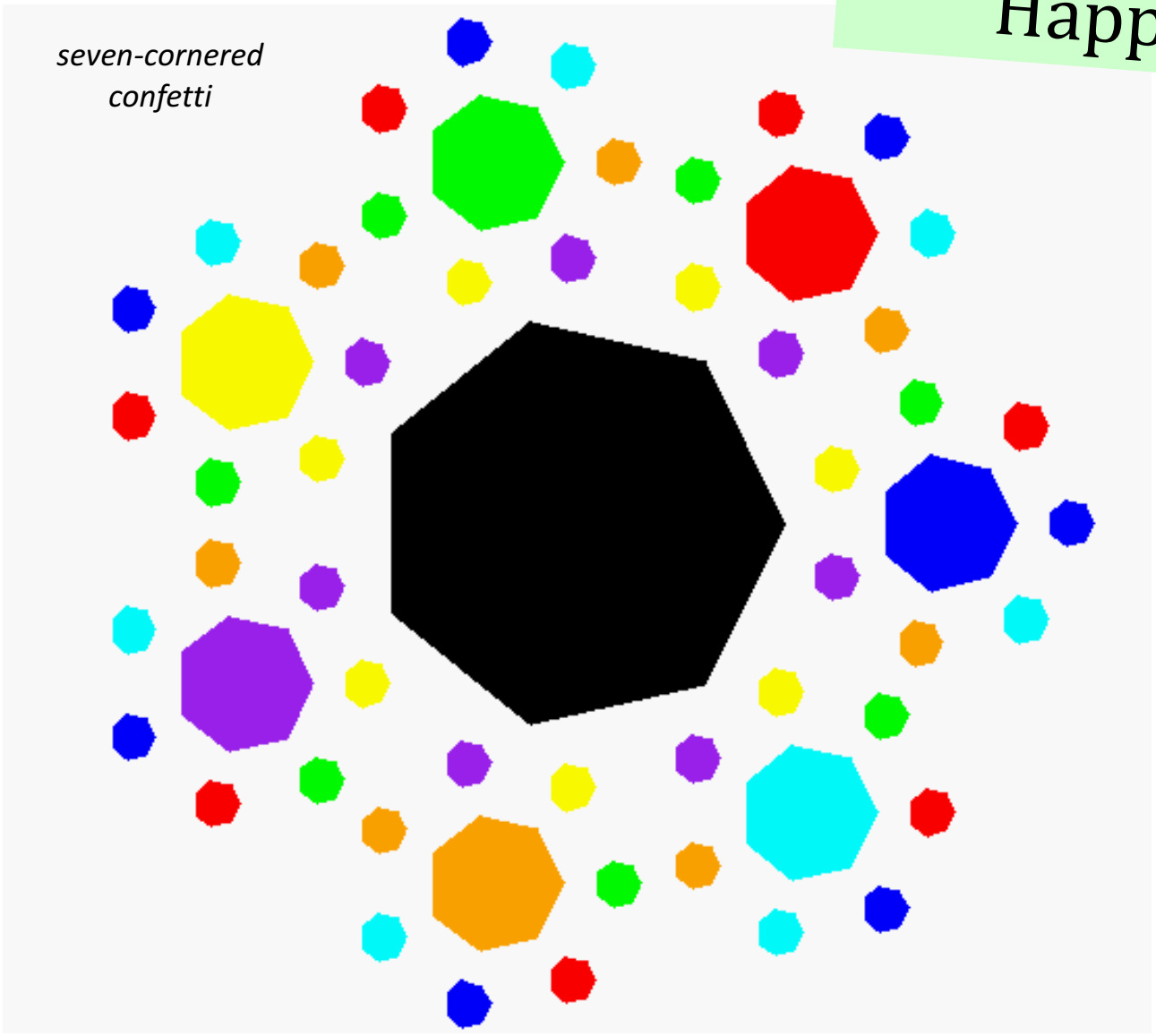


`snowflake(100, 4)`



`snowflake(100, 5)`

Recursive art? Create your own…



Happy turtling!

